Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Peter Hlísta

Algoritmy pro řízení výtahů

Katedra aplikované matematiky

Vedoucí bakalářské práce: RNDr. Josef Cibulka

Studijní program: Informatika

Studijní obor: programování

Praha 2011

Ďakujem svojmu vedúcemu RNDr. Josefovi Cibulkovi za cenné a užitočné rady, drobné návrhy a pripomienky pri vytváraní a ladení programu. Tiež chcem poďakovať za zaujímavé rady, ktoré mi pomohli s vypracovaním Bakalárskej práce.
Chcem sa na tomto mieste aj poďakovať mojej mame, ktorá mi poskytovala psychickú podporu pri vytváraní tohto diela.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne, výhradne s použitím citovaných prameňov, literatúry a ďalších odborných zdrojov. Súhlasím so zapožičiavaním práce a jej zverejňovaním.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Zb., autorského zákona v platnom znení, najmä skutočnosť, že Univerzita Karlova v Prahe má právo na uzavretie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 ods. 1 autorského zákona.

V Prahe dňa 12.7.2011 Peter Hlísta

Název práce: Algoritmy pro řízení výtahů

Autor: Peter Hlísta

Katedra / Ústav: Katedra aplikované matematiky

Vedoucí bakalářské práce: RNDr. Josef Cibulka, Katedra aplikované matematiky

Abstrakt: Práca porovnáva rôzne systémy riešenia výťahov vo výškových budovách s použitím počítačovej simulácie. Tieto systémy sú porovnávané podľa rôznych kritérií, pričom základným kritériom je priemerný čas čakania na výťah. Ďalšími kritériami sú náklady na dopravu a najdlhší čas čakania na výťah. Použité algoritmy sú testované na väčšom počte ľudí než vo väčšine dostupnej literatúry, ktorá skúma zlepšenia väčšinou na dvoch, troch ľuďoch. Výsledky ukazujú, že má zmysel zaoberať sa každým testovaným algoritmom, pretože každý z nich má svoje výhody a nevýhody.

Klíčová slova: výťahy, algoritmy výťahov, simulácia výťahov, výšková budova

Title: Elevator scheduling algorithms

Author: Peter Hlísta

Department: Department of Applied Mathematics

Supervisor: RNDr. Josef Cibulka, Department of Applied Mathematics

Abstract: This thesis compares various systems of elevator solutions in high-rise buildings using computer simulation. These systems are compared according to the various criteria. The basic criterion is the average waiting time for the elevator. The other criteria are the costs of the transport and the longest waiting time. The algorithms for the management of the various types of elevators are tested on the larger group of people than in the most of the available literature that examine the improvements mostly on two or three persons. The results show that it is meaningful to consider all algorithms because each of them has its advantages and disadvantages.

Keywords: elevators, elevator algorithms, elevator simulation, high-rise building

Obsah

1 Úvod

1.1 História výťahov

História výťahov súvisí s vynálezom kladky. Na začiatku ich poháňal človek ručne, ak potreboval niečo vytiahnuť do výšky alebo spustiť do hĺbky. Výťah sa vtedy pohyboval medzi dvoma bodmi v priestore. Postupne sa navrhovali sofistikovanejšie výťahy, či už z hľadiska poháňania (parou, elektrinou), alebo bezpečnosti (pridanie poistky proti pádu). Dané výťahy sa už pohybovali medzi viacerými bodmi v priestore a začali si vyžadovať nejaké stratégie na ich riadenie.

Výťahy delíme podľa rôznych kategórií. Podľa toho, čo sa v nich prepravuje, ich rozdeľujeme v základe na výťahy osobné a výťahy nákladné – či už ide o automobily, stavebný materiál alebo skladovaný tovar. V tejto práci sú rozoberané len osobné výťahy, pretože len ľuďom vadí čakanie.

1.2 Osobné výťahy

Nie všetci ľudia sú dostatočne fyzicky vybavení, aby chodili po schodoch, preto sa začali používať osobné výťahy. V súčasnosti sa s prepravou osôb výťahom počíta vo všetkých vyšších budovách, či už obytných alebo polyfunkčných. Keďže výťahu trvá nejakú dobu, kým prepraví ľudí na želané poschodie, ďalší ľudia musia čakať. Toto čakanie nemôže presiahnuť nejaký rozumný čas. S problémom naprojektovania správneho počtu a typu výťahov na výšku budovy a kapacitu prepravených osôb sa stretávajú stavitelia mrakodrapov.

1.3 Návrh vhodného výťahu

Najjednoduchším spôsobom, ako skrátiť čas čakania ľudí na výťah, je jeho zrýchlenie a zvýšenie jeho nosnosti, teda maximálneho počtu prepravovaných osôb. Tento spôsob môže byť ekonomicky náročný a dosť rýchlo narazíme na strop možností. Pozrime sa však na možnosti riadiť výťahy a vybavovať požiadavky inteligentne pomocou matematických algoritmov.

Počítačová simulácia je výborný spôsob, ako nájsť najlepšie riešenie problému bez toho, aby sme na riešenie problému museli vynaložiť veľké finančné či

časové investície. V práci sa pomocou simulácie testuje niekoľko algoritmov a výsledky sú štatisticky spracované.

Pomocou viacerých druhov algoritmov sa vyhodnocuje čo najefektívnejší spôsob obsluhy prichádzajúcich požiadaviek. Priemerná dĺžka čakania sa štatisticky vyhodnotí pre jednotlivé algoritmy. Jednotlivé časy čakania sú udávané v sekundách, menšie časové jednotky z pohľadu človeka nemajú zmysel, lebo ich nepostrehne.

V niektorých informačných zdrojoch (Benjamin Hiller v prezentácií Online optimalization[2], Ohhoon Kwon, Hyokyung Bahn a Kern Koh v článku A Context-Aware Elevator Scheduling System for Smart Apartment Buildings[1]) sa uvádza, že ak je použitý špecializovaný algoritmus, čas čakania sa vo všeobecnosti skráti, teda ľudia sú obslúžení rýchlejšie. Cieľom tejto práce je zistiť, nakoľko toto tvrdenie zodpovedá prevádzke v praxi. V týchto zdrojoch boli uvedené len teoretické príklady ako jednotlivé algoritmy skrátia dobu čakania pre 2-3 ľudí. V práci sa porovnáva, ako sa skráti čas čakania v budove s mnohými osobami, kde sú výťahy často používané.

V programe je implementovaných sedem rôznych algoritmov pre riadenie výťahu. Päť z nich sa používa v existujúcich budovách, v programe sú testované s menšími úpravami, ktoré sú popísané pri jednotlivých algoritmoch. Dva z nich sú experimentálne algoritmy inšpirované zdrojmi, ktoré sú uvedené v časti 3.4 Algoritmy výťahov.

Pre simuláciu bolo ešte potrebné vytvoriť sadu testovacích súborov. Pomocou generátora sa budú vytvárať požiadavky osôb na prepravu ako informácia o čase, kedy vznikla požiadavka na prepravu, a z ktorého poschodia na ktoré treba osobu prepraviť. Tieto údaje vytvorené generátorom zapíše program do súboru, ktorý bude slúžiť ako vstup do simulácie.

Program obsahuje aj grafickú časť, v ktorej sa zobrazuje vlastný pohyb výťahov s prepravovanými osobami.

V práci sa často vyskytuje termín algoritmus výťahu. V niektorých prípadoch by sa asi zišlo písať typ výťahu, ale keďže dané typy výťahov musia byť a sú riadené nejakým algoritmom, tak označenie algoritmus výťahu namiesto typ výťahu je adekvátne.

2 Ovládanie programu

2.1 Vstupné parametre

Aj generátor aj simulácia vyžadujú k svojmu spusteniu vstupné parametre v príkazovom riadku.

Simulácia

Hlavná časť programu je konzolová aplikácia. Ak nedostane správny počet a typ vstupných parametrov, tak vypíše návod aj s príkladom, ako spúšťať program, a ukončí sa.

K spusteniu programu treba zadať dva alebo tri parametre, a to:

1. Jedno slovo, a to buď „-statistiky“, alebo „-grafika“. Pri zvolení „-statistiky“ sa zvolí simulácia, ktorá vypočíta štatistické údaje a na konci programu ich zapíše do súborov „statistiky_vytahov.txt“ a „statistiky_ludi.txt“. Nastavenie „-grafika“ spustí simuláciu, ktorá sa vykresľuje do nového okna. Okno sa objaví po spracovaní vstupných dát. Ide o povinný parameter.
2. Zadanie relatívnej alebo absolútnej cesty k súboru so vstupnými dátami pre výpočet štatistických údajov. Špecifikácia formátu daného súboru je popísaná nižšie. Ak daný súbor neexistuje, je prázdny, alebo obsahuje nekompletné dáta, program sa ukončí a vypíše, či súbor existuje, či je prázdny, alebo ktorá časť dát mu chýba. Taktiež ide o povinný parameter.
3. Zadanie relatívnej alebo absolútnej cesty k súboru, do ktorého pôjdu výstupné informácie o tom, ako prebieha simulácia. Ide o voliteľný parameter a pri jeho nezadaní sa vypisujú informácie na konzolu. Pri nastavení „-grafika“ a zadaní súboru bude súbor po prebehnutí simulácie prázdny.

Na začiatku programu sa na konzolu vypíšu zadané cesty k vstupnému a výstupnému súboru. Následne sa spracúvajú dáta zo vstupného súboru. Od ich množstva závisí, ako dlho trvá táto fáza. Po spracovaní sa spustí daná simulácia. Na konzolu sa priebežne vypisuje čas simulácie. Pri nastavení „-statistiky“ sa vypisuje aj počet zostávajúcich ľudí v simulácií.

Simulácia sa nespustí v prípade nastavenia „-grafika“, keď program čaká na spustenie simulácie stlačením klávesu „S“ v grafickom okne.

Formát vstupného súboru

Vstupný súbor musí dodržiavať pevne zadaný formát. V súbore sa nikde nepoužívajú medzery a čísla sa používajú vždy celé (patriace do množiny celých čísel). Dáta sú v súbore v textovej podobe a vizuálne tvoria tri časti, ktoré sú medzi sebou oddelené prázdnym riadkom:

1. Časť poschodí obsahuje na prvom riadku číslo väčšie ako 0 a menšie ako 101. Určuje počet poschodí, ktoré sú číslované od 0 do 99. Budova neobsahuje poschodia pod nultým. Z hľadiska štatistických údajov o výťahoch to nie je potrebné. Poschodia sa totiž vždy dajú posunúť tak, aby začínali od nultého. Na nasledujúcich riadkoch je definícia maximálneho počtu ľudí na poschodie. Dá sa zadefinovať viacero poschodí na jednom riadku. Riadok definície poschodí vyzerá tak, že naľavo od „=“ sú zadané čísla reprezentujúce poschodia a napravo je zadané číslo reprezentujúce maximálny počet ľudí na poschodie. Reprezentácia viacerých poschodí je medzi sebou oddelená čiarkou, alebo spojená pomlčkou, čo predstavuje skupinu poschodí od čísla poschodia naľavo po číslo poschodia napravo. Poschodia, ktoré nebudú mať v tejto sekcii zadefinovaný maximálny počet ľudí na poschodie, dostanú v programe pridelenú pevne danú hodnotu 0. Poschodia zadané mimo rozsah poschodí budovy sú ignorované.
2. Časť výťahov obsahuje na prvom riadku číslo reprezentujúce počet výťahov v budove. Pretože ide o simuláciu výťahov, nemôže byť dané číslo menšie ako 1. Maximálne odporúčaný počet výťahov je 50, pretože čas výpočtu simulácie je pri počte výťahov blížiacich sa k 50 príliš dlhý a 50 výťahov v stoposchodovej budove je predimenzovaných. Taktiež grafická simulácia dokáže zobraziť najviac 42 výťahových šácht. Na nasledujúcich riadkoch sú definície výťahov. Definícia výťahu pozostáva z piatich nezáporných celých čísel oddelených medzi sebou dvojbodkami. Prvé číslo predstavuje identifikátor výťahu. Vyjadruje poradie výťahu, prvý výťah bude mať identifikátor 1, druhý 2, atď. Druhé číslo reprezentuje najspodnejšie poschodie v budove, kam až výťah môže dôjsť. Tretie číslo reprezentuje najvrchnejšie poschodie v budove, kam sa výťah dostane. Ak je tretie číslo

menšie ako druhé, tak si ich program sám navzájom vymení. Zároveň si obe čísla upraví tak, aby boli v rozmedzí *<0, počet poschodí budovy - 1>*. Štvrté číslo reprezentuje nosnosť výťahu. Udáva sa v počte ľudí, o ktorých sa predpokladá, že majú rovnakú hmotnosť. Posledné piate číslo priraďuje výťahu algoritmus. Môže to byť jedna zo siedmych možností: 1, 2, 3, 4, 5, 6, alebo 7. Tieto možnosti sú rozobraté v ďalšom texte.

3. Posledná časť je časť o osobách. Nachádzajú sa tu definície osôb, pričom platí, že každý riadok definuje jednu osobu. Pri osobe nás zaujíma, kedy a kam sa potrebuje prepraviť. Preto definícia osoby obsahuje zoznam dvojíc údajov. Prvý údaj je čas a druhý číslo poschodia. Tieto dve informácie sú oddelené čiarkou. Takáto dvojica údajov je opäť oddelená čiarkou od ďalšej dvojice. Definícia času má tvar hodina dvojbodka minúta a udáva, kedy sa človek potrebuje prepraviť na poschodie uvedené za čiarkou.

Program načítava ľudí len dovtedy, kým nenájde znak konca súboru alebo prázdny riadok. Všetky informácie za prázdnym riadkom po časti osoby sa považujú za komentár.

Ovládanie grafického okna

Grafické okno sa otvorí v rozlíšení 512x512 pixelov v ľavom hornom rohu obrazovky. Šírka a výška okna sa dá ľubovoľne nastavovať, pričom minimálna šírka je 104 pixelov a minimálna výška je 1 pixel. Okno sa môže prepínať medzi zobrazením na celú obrazovku alebo oknom. K ovládaniu okna sa používa klávesnica alebo myš. Zadanie veľkého písmena alebo malého písmena aplikácia nerozlišuje.

Popis ovládania myšou:

Grafické zobrazenie sa ovláda pomocou myši podobne ako v mnohých 3D-zobrazovacích programoch. Pri použití kolieska myši nahor alebo nadol sa budova približuje alebo vzďaľuje v určitých medziach. Ďalšími možnosťami je pohyb myšou so stlačeným ľavým tlačidlom na myši. Pri pohybe myšou nahor alebo nadol sa budova zobrazuje pod uhlom od 0 do 80 stupňov voči podlahe budovy. Pri pohybe myšou doľava alebo doprava sa budova otáča okolo svojej osi.

Zoznam písmen a tlačidiel klávesnice, na ktoré aplikácia reaguje:

Kláves:	Funkcia:
S	Najdôležitejšie tlačidlo. Spúšťa a pauzuje simuláciu.
Q	Vypína grafické okno, a tým aj celý program.
F	Prepína aplikáciu medzi oknom a celou obrazovkou.
R	Nastavuje okno na počiatočnú veľkosť okna a pozíciu na ploche.
W	Tá istá funkcia ako tlačidlo „R“.
ESC	Tá istá funkcia ako pri „Q“.
šípka nahor	Približuje pohľad na budovu až po určitý bod.
šípka nadol	Vzďaľuje pohľad na budovu až po určitý bod.
šípka doprava	Otáča budovu okolo jej osi. Ide o pohyb proti smeru hodinových ručičiek.
šípka doľava	Otáča budovu okolo jej osi. Ide o pohyb v smere hodinových ručičiek.
page up	Posúva budovu o jedno poschodie nadol. Maximálne však po strechu budovy.
page down	Posúva budovu o jedno poschodie nahor. Maximálne však po podlahu budovy.
home	Nastaví počiatočný pohľad na budovu.
end	Nastaví pohľad na strechu mrakodrapu s informačným výpisom o stave simulácie.

Tabuľka 2.1: Ovládanie grafického okna

Pri uzavretí okna sa simulácia ukončí.

Generátor

Súčasťou programu je aj generátor. Ide o konzolovú aplikáciu. Vstupné parametre sú oproti parametrom simulácie jednoduchšie. Pri spustení generátora v príkazovom riadku je potrebné zadať len dva parametre. Ak nie sú zadané, vypíše sa návod aj s príkladom ako spúšťať program a ukončí sa.

Popis parametrov generátora:

1. Zadanie relatívnej alebo absolútnej cesty k súboru so vstupnými dátami pre vygenerovanie výstupného súboru. Je to povinný parameter.
2. Zadanie relatívnej alebo absolútnej cesty k súboru, kam pôjdu výstupné dáta. Ak daný súbor neexistuje, vytvorí sa. Ak už existuje, vymaže sa jeho obsah. Ide o povinný parameter.

Výstupný súbor generátora slúži ako vstupný súbor simulácie. Generátor teda upraví a doplní svoj vstupný súbor tak, aby zodpovedal podmienkam vstupu simulácie. Podrobne je činnosť generátora popísaná v časti 3.3 Popis činnosti programu.

Formát vstupného súboru

V prvom riadku je celé kladné číslo menšie ako 101. Definuje počet poschodí. V druhom riadku je opäť celé kladné číslo, ktoré určuje počet osôb žiadajúcich o prepravu. Odporúčané obmedzenie pre ich počet je 5000, pretože čas simulácie sa začne neprijateľne predlžovať.

Počet výťahov je definovaný v treťom riadku. Na nasledujúcich riadkoch sú definície jednotlivých výťahov rovnaké ako pri simulácii popísané vyššie v časti Simulácia – Formát vstupného súboru.

2.2 Výstup programu

Jednotlivé časti programu majú vlastné výstupy. Ide hlavne o výstupy na konzolu a do súborov. Existuje aj možnosť grafického výstupu do samostatného okna.

Simulácia

Nastavenie „-statistiky“ spôsobí vytvorenie troch súborov. Počas činnosti programu sa informácie vypisujú na konzolu. V prvých dvoch riadkoch sa vypíše cesta k vstupnému a výstupnému súboru. V ďalšom riadku sa aktualizujú informácie o aktuálnom čase a počte prepravovaných osôb. Na konci programu sa na ďalšie riadky vypíšu dve správy o zapisovaní štatistických údajov do dvoch súborov.

Pri nastavení „-grafika“ sa vypíše na konzolu informácia o zadaných parametroch. Vytvorí sa nové grafické okno, bližšie informácie sú nižšie v časti

Obrazovka grafického výstupu. V ďalšom riadku sa aktualizujú informácie o aktuálnom čase simulácie. Po ukončení sa vypíše správa o jej konci.

Formát výstupného súboru zo simulácie

Na každom riadku výstupného súboru sa nachádza jedna informácia zo simulácie v tvare:

- čas – hodina dvojbodka minúta, čísla sú písané dvojciferne
- objekt – človek alebo výťah a jeho identifikátor
- čo vykonal – vypíše sa niektorá z nasledujúcich správ: „človek vošiel do budovy“, „človek sa snaží dostať na poschodie“, „človek sa postavil do rady hore na výťah na poschodí“, „človek sa postavil do rady dole na výťah na poschodí“, „človek vystúpil z výťahu na poschodí“, „človek dorazil na posledné poschodie cesty“, „človek odpočíva do“, „výťah sa zobudil a otvára dvere na poschodí“, „výťah sa zobudil a ide hore“, „výťah sa zobudil a ide dole“, „výťah otvoril dvere a ľudia vystupujú a nastupujú“, „výťah zatvára dvere s počtom ľudí“, „výťah bol zavolaný na poschodí a ide hore“, „výťah bol zavolaný na poschodí a ide dole“, „výťah došiel na poschodie a otvára dvere“, „výťah sa ukladá na spánok“

Formát súboru štatistických údajov o výťahoch

Súbor štatistických údajov má dve časti. V prvej je v každom riadku napísané, o ktorý výťah ide a jeho štatistické údaje:

- priemerný čas čakania – predstavuje aritmetický priemer všetkých časov čakania osôb na daný výťah
- maximálny čas čakania – čas, koľko osoba najdlhšie čakala na daný výťah
- energetická spotreba výťahu – koľko elektriny spotreboval daný výťah na svoje fungovanie, ide o abstraktnú premennú, ktorá sa vypočíta z počtu prejdených poschodí, z počtu prepravovaných osôb a z počtu otvorení a zatvorení dverí. Jedna jednotka spotreby je stanovená ako spotreba elektriny na prejdenie prázdneho výťahu o jedno poschodie. V prípade, že ide výťah nahor, spotreba sa vynásobí koeficientom 1,1, v prípade, že ide nadol, koeficientom 0,9. Pre každú prepravovanú osobu sa spotreba vynásobí koeficientom 1 + (0,05 * počet prepravovaných osôb).

- vyťaženosť výťahu – dáva do pomeru počet poschodí, v ktorých prešiel daný výťah prázdny k počtu poschodí, v ktorých výťah viezol aspoň jedného človeka

Druhá časť súboru obsahuje jeden riadok, v ktorom sú štatistické údaje o všetkých výťahov dohromady a celková spotreba elektriny všetkých výťahov v budove.

Formát súboru štatistických údajov o osobách

Súbor štatistických údajov o osobách má tiež dve časti. Na začiatku sú samostatné štatistické údaje o každej osobe, a to:

- priemerný čas čakania – aritmetický priemer jednotlivých časov čakania danej osoby na hociktorý výťah
- maximálny čas čakania – najdlhší čas čakania zo všetkých časov čakania danej osoby
- priemerný čas jazdy – aritmetický priemer času, ktoré človek strávil jazdou vo výťahu
- maximálny čas jazdy – najdlhší čas, ktorý strávil človek jazdou vo výťahu

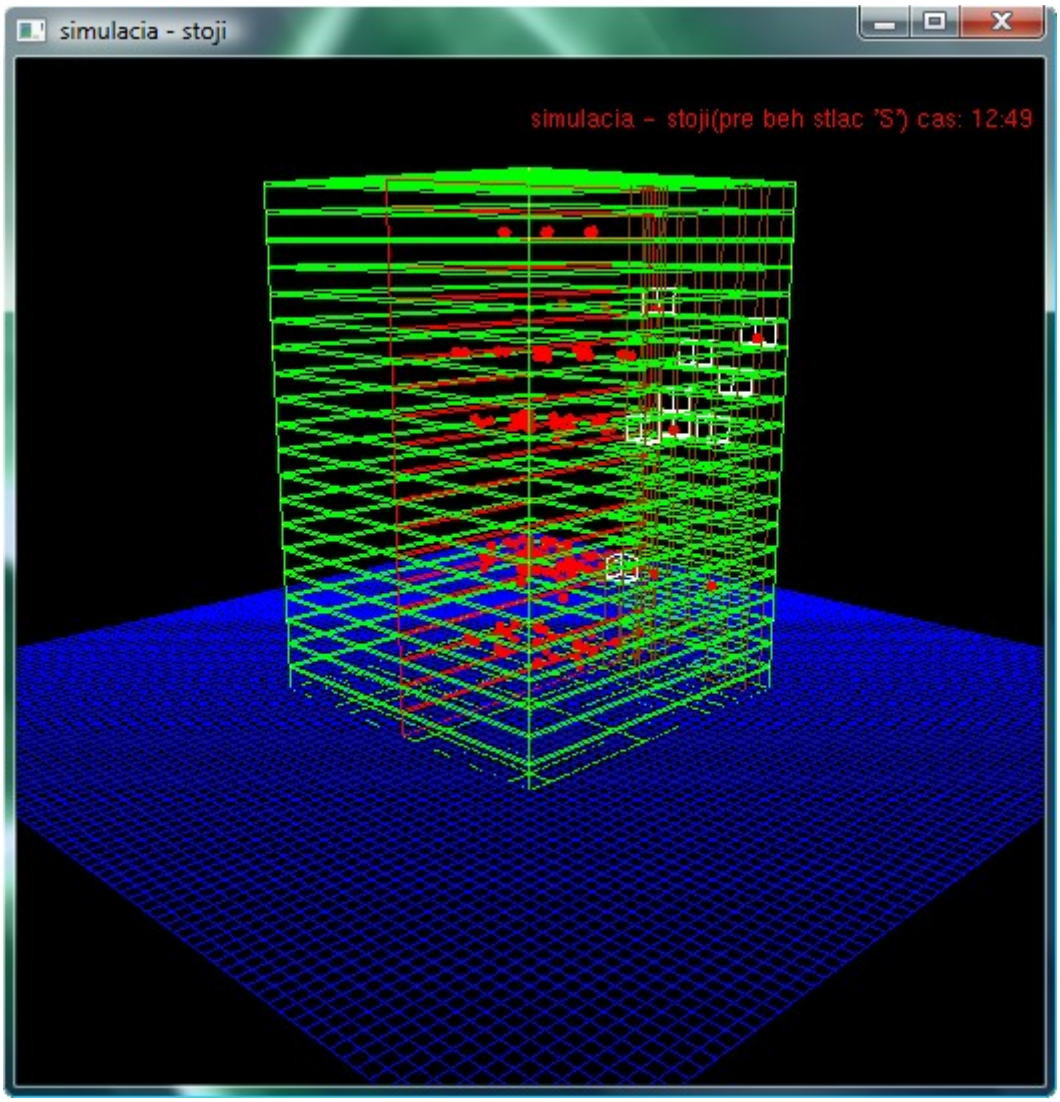
Druhá časť obsahuje súhrnné štatistické údaje o všetkých osobách uvedené v jednom riadku, a to:

- priemerný čas čakania na výťah všetkých osôb v budove
- najdlhší čas čakania na výťah všetkých osôb v budove

Obrazovka grafického výstupu

Legenda k obrázku 2.1:

- modrá mriežka tvorí podlahu scény
- zelené čiary znázorňujú vonkajšie hrany budovy, strechu budovy, jednotlivé poschodia a nástupný priestor pred výťahom
- hnedé sú šachty pre výťahy
- biele sú kabínky výťahov
- červené obdĺžniky oddeľujú sekciu pre ľudí a sekciu pre výťahy v budove na poschodiach a červené guľôčky reprezentujú ľudí



Obrázok 2.1: Okno grafického výstupu

Generátor

Výstup sa zapisuje do súboru a na konzolu. Na konzolu sa vypisujú informácie, čo práve v danej chvíli robí generátor, či generuje človeka alebo zapisuje do súboru. Do súboru sa zapisujú dáta, ktoré sa neskôr použijú ako vstup pre simuláciu. Dáta sa môžu ešte trochu upraviť. Výstup generátora má formát vstupného súboru pre simuláciu.

2.3 Informácie k spusteniu programu

Program je vytvorený pre systémy Windows. K spusteniu programu je potrebná dynamicky linkovaná knižnica freeglut.dll. Dá sa získať na stránke http://freeglut.sourceforge.net/. Daná knižnica freeglut.dll je zatiaľ k dispozícii len pre 32-bitový systém Windows, preto na niektorých 64-bitových systémoch Windows tento program nepobeží.

Program nebol vytvorený pre unixové systémy. Po malých úpravách ho asi je možné použiť aj v systémoch Unix. Je však potrebné si stiahnuť knižnicu freeglut, skompilovať ju na danom systéme a pridať k zdrojovým súborom. Pri vytváraní programu je potrebné ešte nezabudnúť, že kompilátor potrebuje vedieť, kde sa nachádzajú hlavičkové súbory knižnice a linker potrebuje vedieť adresár, kde sa nachádza knižnica freeglut.a. Na následné spustenie programu v systéme Unix treba mať vytvorenú knižnicu freeglut.so.

3 Implementácia

3.1 Návrh simulácie

Jednou z prvých vecí v simulácii, ktoré bolo potrebné vyriešiť, bolo vkladanie objektov do dátovej štruktúry ako budúce udalosti. Do úvahy prichádzali halda alebo obojstranný spojový zoznam. U obojstranného spojového zoznamu znamená pridanie nového prvku zložitosť O(n), získanie minima znamená konštantnú zložitosť a zmazanie minima je tiež konštantná zložitosť. U haldy pridanie prvku znamená zložitosť O(log n), získanie minima znamená konštantnú zložitosť a zmazanie minima znamená zložitosť O(log n). V každom kroku simulácie sa vyberie minimum zo štruktúry, zmaže sa a do štruktúry sa vloží späť upravený prvok. Pre haldu to znamená zložitosť O(log n) a pre spojový zoznam O(n). Z toho jednoznačne vychádza halda ako lepšie riešenie. V programe je použitá reprezentácia haldy ako poľa.

Ďalšou vecou, ktorú bolo potrebné vyriešiť, bolo naplnenie haldy. Bolo potrebné simulovať výťahy a zároveň osoby. Jednou z možností bolo dať objekty do dvoch oddelených štruktúr, vždy vybrať minimum z oboch a tie ďalej porovnávať, alebo ich dať do jednej štruktúry a spraviť z osôb a výťahov potomkov nejakej virtuálnej triedy. Obyčajne sa tento typ problému rieši dedičnosťou, čo sa ukázalo ako vhodné riešenie aj v tomto prípade. Predkom je trieda udalost.

Ďalším problémom bolo určenie, ktorý objekt sa bude spracovávať ako prvý. Rozhodujúcim kritériom sa ukázala premenná čas. Tu sa vynorila otázka, či to bude iba jednoduché číslo, ktoré bude reprezentovať počet sekúnd od začiatku simulácie, alebo sa na to vytvorí samostatná trieda. Pre jednoduchšie narábanie s premennou sa zvolila samostatná trieda.

Trieda človek

Je to prvá dôležitá trieda, v ktorej sa vykonáva logika simulácie. Pri návrhu simulácie sa predpokladalo, že každá osoba bude jednať poctivo, teda použije výťah, ktorý si privolala ona a nie niektorá iná osoba, hoci prišiel skôr. Nebude privolávať viacero výťahov naraz a nebude pozerať na to, koľko ďalších osôb už čaká na konkrétny výťah.

Trieda človek obsahuje ďalšie triedy:

- cesta – obsahuje požiadavky osoby na prepravu. Keď sa dôjde na koniec ich zoznamu, osoba prestáva byť pre simuláciu zaujímavá a je vyradená zo simulácie.
- cesty – podáva informáciu, ktorými výťahmi sa osoba môže prepraviť z poschodia A na poschodie B. Priradenie konkrétnej možnosti prebieha náhodne.
- statistika_cloveka – zaznamenávajú sa tu štatistické údaje ohľadom času čakania človeka na výťah a času stráveného vo výťahu.

Trieda výťah

Je to najdôležitejšia trieda simulácie. Sú v nej zahrnuté všetky algoritmy výťahov. Trieda sa stará o riadenie radov na výťah, aktualizovanie informácií o osobe pri nastupovaní do výťahu a pri vystupovaní. Obsahuje v sebe triedu statistika_vytahu, do ktorej sa zaznamenávajú štatistické údaje o výťahu. Obsahuje tiež ukazovatele na osoby, ktoré sú aktuálne vo výťahu a ktoré ešte čakajú v rade na výťah. Môže vďaka tomu aktualizovať informácie napríklad o dĺžke ich čakania na výťah a čísle poschodia, kde vystúpili.

Bolo tu spomenutých niekoľko tried, ktoré ešte neboli popísané. Začneme triedou cesta. Ide o triedu, ktorá spája dvojice čas - poschodie do zoznamu. Ďalej bola spomenutá trieda cesta_vytahom. Je podobná triede cesta, pretože ide o zoznam dvojíc. V tomto prípade však ide o dvojice poschodie – ukazovateľ na výťah. Definuje, ktorý výťah má osoba použiť a na ktoré poschodie má ísť. Postupným spracovaním tohto zoznamu sa osoba prepraví z poschodia, kde pôvodne je, na poschodie, kam sa chce dostať. Posledné dve spomenuté triedy sú statistika_clovek a statistika_vytah. V nich sa zhromažďujú a počítajú štatistické údaje o jednotlivých objektov a na konci programu sa zapíšu do súboru.

Grafická simulácia

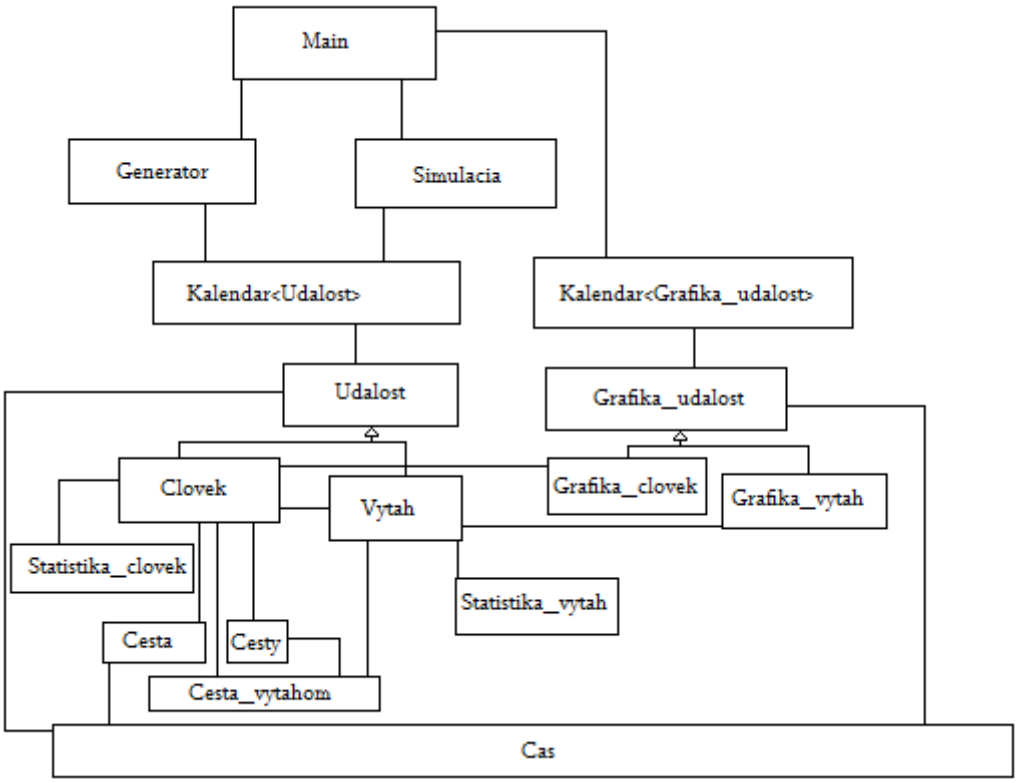
Ku grafickému zobrazeniu simulácie bola potrebná grafická knižnica s jednoduchým *application programming interface (API)*. Ako vhodná voľba sa ukázala grafická knižnica freeglut, ide o pokračovateľa grafickej knižnice GLUT, ktorá je nadstavbou nad OpenGL.

Vyskytol sa tu jeden problém. Grafická simulácia nemôže byť naprogramovaná do triedy, pretože freeglut vyžaduje funkcie s presne danými parametrami a v c++ volanie funkcie z triedy má ako prvý parameter ukazovateľ na danú triedu, čo globálne funkcie nemajú. Preto bolo potrebné celú grafickú simuláciu presunúť z triedy simulácia do funkcie main.

Aby všetko nebolo v súbore main.cpp, bolo potrebné vytvoriť ďalšie triedy, zahrňujúce metódy na vykreslenie objektov človek a výťah. Vzhľadom na to, že to už boli rozsiahle triedy, sa to javí ako správne rozhodnutie. Preto boli vytvorené triedy grafika_clovek a grafika_vytah a dostali ukazovateľ na pôvodné triedy. Vďaka tomu riešia iba vykresľovanie objektu do okna. Aby ich bolo možné pridávať do haldy, bolo potrebné prepísať triedu kalendar na template triedu a vytvoriť predka grafickým triedam grafika_clovek a grafika_vytah.

V grafickej simulácii sa budova zobrazuje ako trojrozmerný objekt. Vyskytla sa tu otázka, ako zobrazovať scénu simulácie. Vďaka tomu, že freeglut má metódu, v ktorej sa zadáva pozícia kamery a bodu, kam sa pozerá (a toho, kde je strop scény), bolo možné si zvoliť, ako ovládať kameru. Boli tri vhodné možnosti: 1. pevne daná statická kamera, 2. kamera umiestnená v okolí budovy pozerajúca sa dovnútra budovy a 3. voľne sa pohybujúca kamera. Ako zaujímavou sa zdala druhá možnosť, lebo sa dá rýchlo meniť pohľad na budovu.

3.2 *Prepojenie objektov*



Obrázok 3.1: Náčrt prepojenia tried programu

Na obrázku 3.1 je znázornené prepojenie jednotlivých tried programu, v nasledujúcom texte sú vysvetlené jednotlivé prepojenia.

Main nie je trieda, ale reprezentuje hlavnú funkciu programu. V nej sa vytvorí buď generátor alebo simulácia podľa toho, na čo je určený program. Najdôležitejšia súčasť oboch tried je trieda kalendar. Z nej sa vyberajú, spracovávajú a prípadne vkladajú späť jednotlivé objekty clovek a vytah.

Clovek a vytah majú spoločného predka, abstraktnú triedu udalost. Tá je prepojená len s časom, pretože podľa času sa porovnáva, ktorá udalosť je skôr na rade.

Vráťme sa späť k triede vytah. Táto trieda je prepojená s triedou statistika_vytahu, ktorá zabezpečuje ukladanie štatistických údajov o výťahu. Ďalej je prepojená s triedou cesta_vytahom, a to preto, že sa v nej používa ukazovateľ na výťah. Prepojenie s triedou clovek je použité preto, aby mohla trieda vytah zavolať metódu triedy clovek, ktorá zaktualizuje údaje. Posledné prepojenie vytah

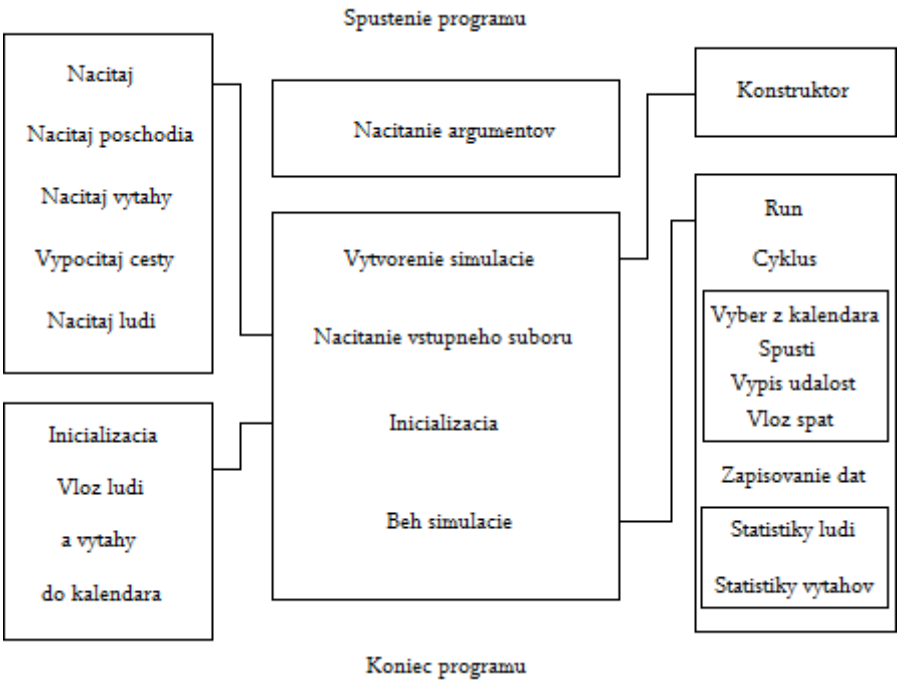
s grafika_vytah je dôležité preto, aby grafika_vytah slúžila len na vykreslenie výťahu a logika výťahu zostala v triede vytah.

Teraz sa pozrime na triedu clovek. Táto trieda je prepojená s triedou cesta, ktorá reprezentuje logiku osoby. Ďalej je prepojená s cesta_vytahom. Tu si zapamätáva údaje z triedy cesty, ako sa prepraviť na želané poschodie. Samozrejme je prepojená s triedou statistika_cloveka, ktorá zhromažďuje štatistické údaje o človeku.

Nachádzajú sa tu ešte triedy určené pre grafickú simuláciu, a to grafika_clovek a grafika_vytah. Obe sa starajú len o vykresľovanie objektu, ktorý reprezentujú. Majú spoločného predka graficka_udalost, ktorý je skoro identickou kópiou triedy udalost. Vďaka tomu je možné ľahko uložiť všetky objekty do haldy. Tá je využívaná aj pri grafickej simulácii.

3.3 Popis činnosti programu

Popis činnosti simulácie so štatistickými údajmi

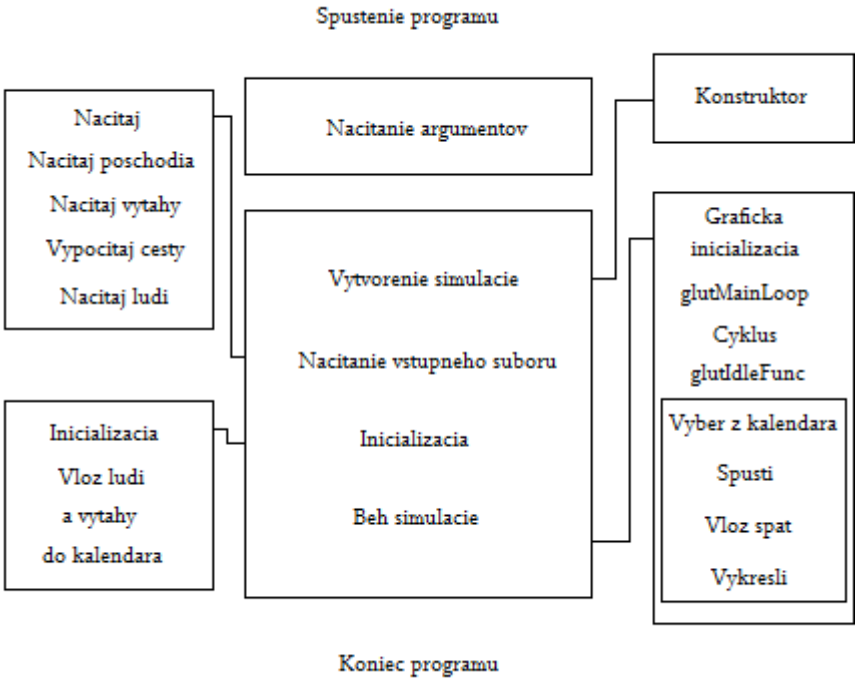


Obrázok 3.2: Simulácia s výstupom štatistických údajov

Na obrázku 3.2 je znázornené, ako sa postupne spúšťajú jednotlivé časti programu a funkcie. Je dôležité si všimnúť, že pri načítavaní vstupného súboru sa ešte pred samotným načítaním osôb zo súboru vypočítajú cesty v budove. To znamená, že sa nájdu najkratšie cesty medzi všetkými dvojicami poschodí cez výťahy.

Zapisovanie štatistických údajov o ľuďoch a výťahoch sú dve privátne metódy, ktoré sa spustia po skončení simulácie. Počas simulácie sa zapisujú udalosti do výstupného súboru alebo na konzolu.

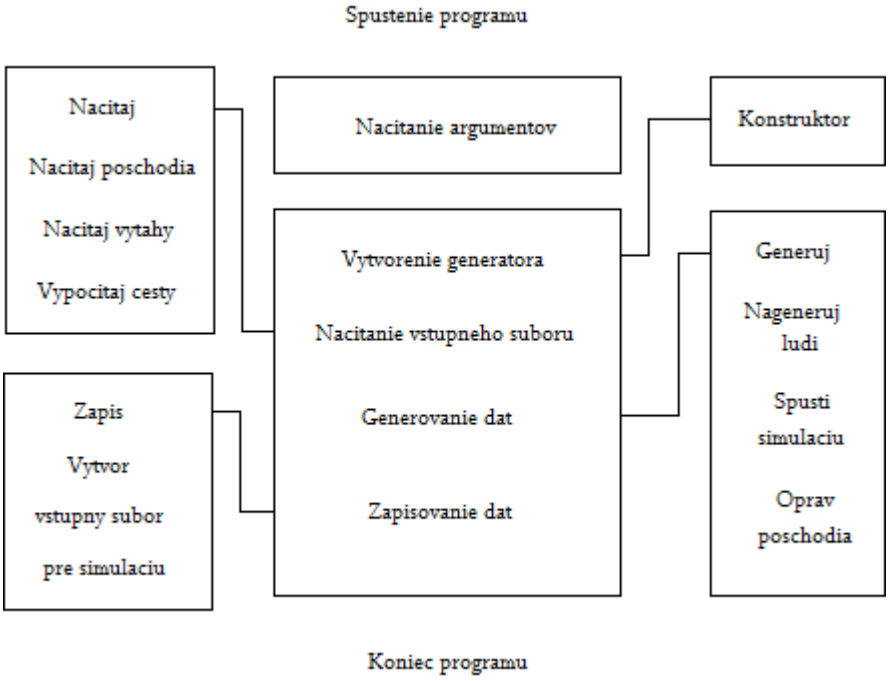
Popis činnosti simulácie s grafickým výstupom



Obrázok 3.3: Simulácia s grafickým výstupom

Obrázok 3.3 sa od obrázku 3.2 veľmi nelíši, pretože obrázok 3.3 zobrazuje volanie funkcií pri nastavení na grafický výstup programu. Jediným rozdielom je beh simulácie. V tomto prípade, kvôli použitiu grafickej knižnice freeglut prebieha samotná grafická simulácia vo funkcií idle, ktorá je registrovaná pomocou funkcie glutIdleFunc.

Popis činnosti generátora



Obrázok 3.4: Generátor

Obrázok 3.4 znázorňuje, ako sa postupne spúšťajú jednotlivé časti programu a funkcie. Na rozdiel od simulácie sa ľudia nenačítavajú, ale generujú.

Generátor pracuje tak, aby daná simulácia modelovala bežnú prevádzku pracovného dňa v podniku. Osoby teda majú pevne stanovené poschodie, kde pracujú, pričom občas náhodne chodia na iné poschodia. V čase raňajok, obeda a večere sa idú najesť na jedálenské poschodie. Dĺžka pracovnej doby, dĺžka jedenia a začiatok pracovnej doby sú tiež generované náhodne pre každú osobu.

Následne sa vygenerované údaje použijú v simulácií, v ktorej sa nepočítajú štatistické údaje ani neprebieha simulácia v grafickom okne. Po jej prebehnutí sú upravené nedostatky s kapacitou poschodí. Posledný krok je vytvorenie vstupného súboru pre simuláciu.

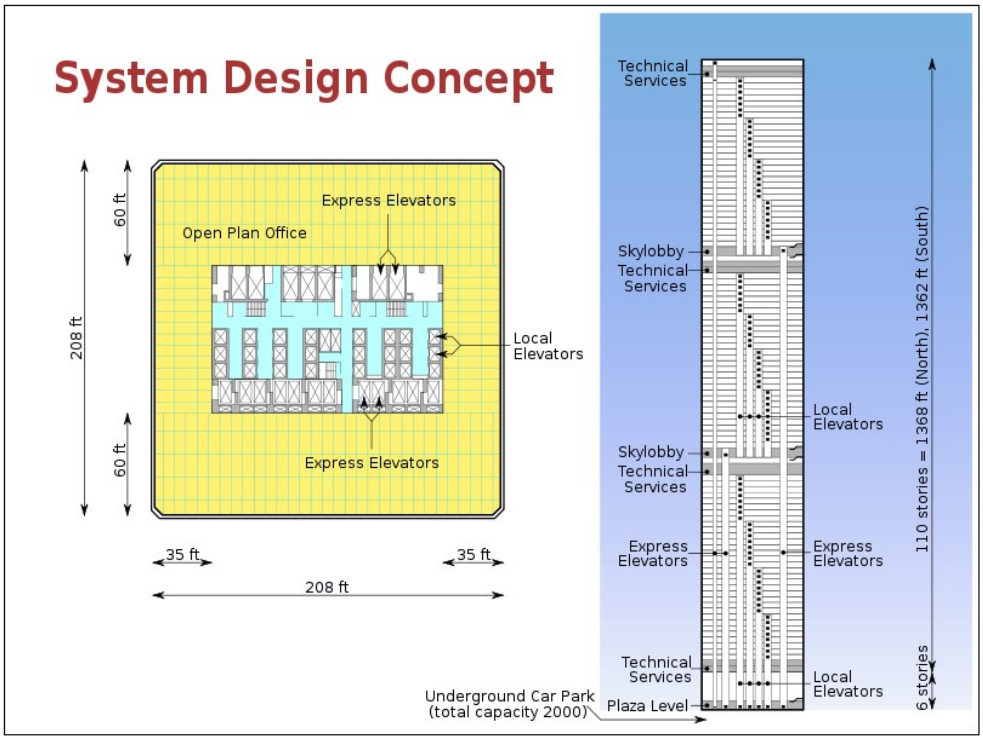
3.4 Algoritmy výťahov

Pre prácu bolo použitých sedem rôznych algoritmov inšpirovaných skutočnými algoritmami, ktoré sa používajú v budovách. Každý algoritmus sa v niečom odlišuje

od ostatných. V simulácii sú dvere výťahu ovládané výťahom a nie ručne osobami. Taktiež výťahy zrýchľujú a spomaľujú nárazovo, čo nie je síce v realite možné, ale pre štatistické údaje, kedy a kde sa daný výťah nachádza, je to postačujúce. Čas potrebný na zrýchlenie a spomalenie je prirátaný k času potrebnému na otvorenie a zatvorenie dverí.

Privolávacie tlačidlá výťahov fungujú vo väčšine prípadov pre smer nadol alebo nahor, pokiaľ to nie je pri danom algoritme popísané inak.

Názvy algoritmov boli vytvorené pre potreby práce, nejde o žiadne oficiálne alebo zaužívané pomenovanie uvedených algoritmov.



Obrázok 3.5: Usporiadanie výťahov v budove WTC

Zdroj: wikipedia.org/wiki/File:World_Trade_Center_Building_Design_with_Floor_and_Elevator_Arrangment.svg

1. Klasický algoritmus

Ide o najčastejšie používaný algoritmus v malých budovách približne do 20 poschodí. Do výťahu sa dá nastúpiť na každom poschodí, to znamená, ak výťah chodí od 0 do 10, tak sa dá zavolať na všetkých 11 poschodiach. Staršie typy týchto výťahov ešte nemajú dvere výťahu ovládané výťahom, ale ručne.

Samotný algoritmus je tiež jednoduchý. Výťah zastaví v smere aktuálneho pohybu na každom poschodí, kam bol zavolaný alebo aj poslaný. Ak už nemá v danom smere poschodia, na ktoré bol zavolaný alebo poslaný, otočí svoj smer, aby vyhovel požiadavkám v opačnom smere. Ak nie je zavolaný na žiadnom poschodí, ostane stáť na aktuálnom poschodí.

Je to základný algoritmus bez heuristík. Používa sa tiež v počítačoch pri čítaní z pevného disku.

2. Rýchly algoritmus

Najväčšia výhoda rýchleho algoritmu je v schopnosti rýchlo prepraviť veľký počet osôb medzi spodným a horným poschodím. Používal sa napríklad v budovách World Trade Center (obrázok 3.5).

Spodné a vrchné poschodia sú špeciálnym typom poschodí, takzvané haly, z ktorých sa dá dostať na jednotlivé poschodia danej časti budovy. Výťah prepravuje len medzi dvoma halami, takže sa môže pohybovať veľkou rýchlosťou. Na ostatné poschodia sa ľudia prepravujú ďalšími výťahmi, ktoré používajú iné algoritmy.

Vďaka tomu je tento algoritmus veľmi jednoduchý.

3. Viacposchodový algoritmus

Aj viacposchodový algoritmus bol použitý v budovách World Trade Center (obrázok 3.5). V podstate ide o upravený klasický algoritmus. Výťah chodí na spodné poschodie, a potom na určitý počet vrchných poschodí, medzi spodným poschodím a vrchnými poschodiami je zopár poschodí, kde výťah nikdy nezastavuje. Výťah je teda možné zavolať len na spodnom a vrchných poschodiach. Samotný algoritmus pracuje ako pri klasickom výťahu.

4. Dvojposchodový algoritmus

Dvojposchodový algoritmus sa používa pre výťah, ktorý má dve kabínky pevne spojené nad sebou (obrázok 3.6). Kvôli obmedzeniu výťahovej šachty sa vrchná kabínka nikdy nedostane na najspodnejšie poschodie, kam výťah chodí, a naopak spodná na najvrchnejšie poschodie. Túto vlastnosť bolo treba implementovať do logiky výťahu. Pri vstupných dverách do výťahu sú z toho dôvodu až štyri privolávacie tlačidlá. Dva smerom nahor, z toho jedno pre najvyššie poschodie a jedno pre ostatné v smere nahor a dva smerom nadol s podobnou logikou.

V praxi sa tiež používa variant, keď jedna z kabín zastavuje na párnych poschodiach a druhá na nepárnych.

Algoritmus daného výťahu funguje podobne ako u klasického algoritmu s rozdielom, že algoritmus skontroluje či danú požiadavku môže splniť len konkrétna časť výťahu (horná alebo spodná kabínka) alebo môže zastaviť hociktorá kabínka.

Výhoda daného algoritmu sa prejavuje v dopravných špičkách, keď je potrebné prepraviť veľké množstvo osôb. Taktiež výťah zaberá len jednu šachtu, čím racionalizuje využitie drahého priestoru v budove. Daný výťah má aj dobré výsledky v šetrení energie.

Obrázok 3.6: Dvojposchodový výťah v budove C.D. Howe v Ottawe v Kanade

Zdroj: wikipedia.org/wiki/File:240_Sparks_Elevators.jpg

5. Dvojitý algoritmus

Ide o algoritmus, keď sú dva výťahy ovládané spoločnými privolávacími tlačidlami. Každý z výťahov pracuje samostatne ako v klasickom algoritme, ale spolu sa rozhodujú, ktorý z nich je schopný rýchlejšie obslúžiť dané zavolanie. V tom spočíva ich výhoda. Napríklad jeden obsluhuje osoby idúce nahor a druhý osoby idúce nadol.

Algoritmus funguje tak, že si vypočíta, ktorý z dvoch výťahov môže skôr obslúžiť aktuálnu požiadavku vzhľadom na svoj aktuálny smer pohybu a ostatné nevybavené požiadavky a vyšle jeden z nich.

6. Predvídavý algoritmus

V dnešnej dobe vznikajú moderné budovy, v ktorých sú senzory na pohyb ľudí, ktoré vedia predpokladať, že nejaká osoba v nasledujúcej chvíli pôjde k výťahu, aby ho použila. Presne pre takúto budovu je určený tento výťah.

Algoritmus výťahu dostane vďaka inteligentnej budove informáciu, že osoba ide privolať výťah ešte predtým, než tak osoba skutočne spraví. Algoritmus teda funguje ako klasický algoritmus s tým rozdielom, že za privolanie výťahu považuje aj informáciu od inteligentnej budovy, že bude privolaný.

V programe dostane výťah informáciu v priemere 5 sekúnd pred stlačením. Výťah tak môže v niektorých prípadoch razantne skrátiť čas čakania. Čas čakania sa môže aj predĺžiť, pretože kým osoba nestačí tlačidlo na privolanie, výťah nevie, ktorým smerom bude chcieť ísť. Viac danú myšlienku rozvíjajú Kwon, Bahn a Koh[1]. Na obrázku 3.7 je znázornený rozdiel fungovania predvídavého a klasického algoritmu. Ide o prekreslený obrázok z uvedeného zdroja.

Majme troch pasažierov:

Prvý pasažier je na poschodí 1 a v čase t = 3 sekundy sa postaví do rady na výťah a chce ísť na poschodie 10.

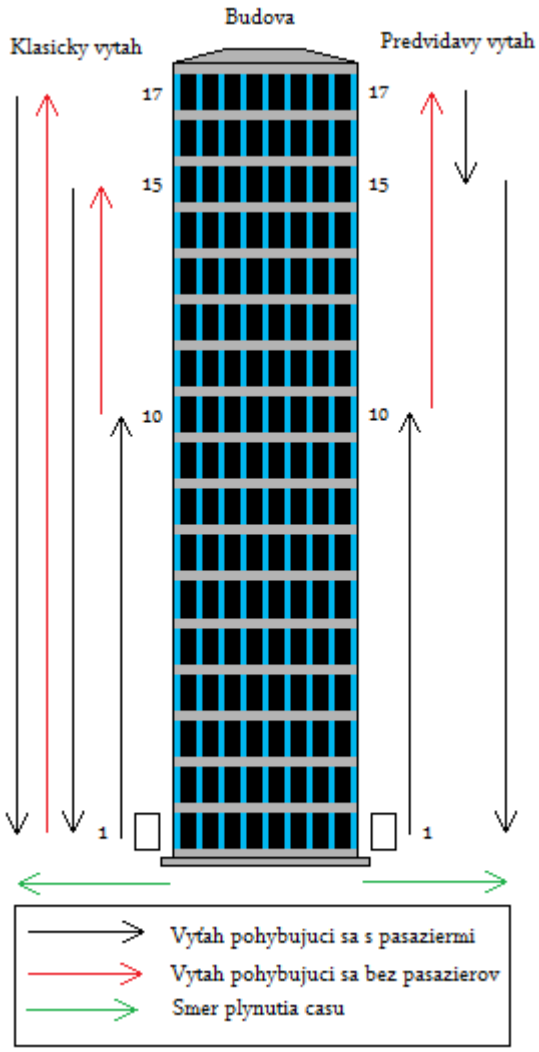
Druhý pasažier je na poschodí 15 a v čase t = 10 sekúnd sa postaví do rady na výťah a chce ísť na poschodie 1.

Tretí pasažier je na poschodí 17 a v čase t = 18 sekúnd sa postaví do rady na výťah a chce ísť na poschodie 1.

Pri klasickom algoritme bude vybavenie vyzerať nasledovne: výťah zoberie prvého pasažiera v čase t = 3 na poschodí 1. Vyjde na poschodie 10 v čase t = 12. Prvý pasažier vystúpi. Následne ide na poschodie 15. Tam je v čase t = 17. Nastúpi druhý pasažier. Výťah zíde na poschodie 1. V čase t = 31 vystúpi druhý pasažier. Následne ide výťah po tretieho pasažiera na poschodie 17. Príde tam v t = 47. Zoberie tretieho pasažiera a zvezie ho na poschodie 1. Skončí v čase t = 63.

Nech predvídavý výťah vie 3 sekundy dopredu, že človek chce použiť daný výťah. Obsluha bude potom vyzerať takto: výťah zoberie prvého pasažiera v čase t = 3 na poschodí 1. Vyjde na poschodie 10 v čase t = 12. Vystúpi prvý pasažier. Ide na poschodie 15. Tam by dorazil v čase t = 17, ale v čase t = 15 sa dozvie, že bude zavolaný na poschodie 19. Keďže druhý pasažier chce ísť nadol, tak výťah ide najprv na poschodie 19. Dorazí tam v čase t = 19. Medzitým už druhý pasažier zavolal daný výťah, takže doň nastúpi. Výťah následne ide na poschodie 15. Príde tam v čase

t = 21 a naberie tretieho pasažiera. Následne zíde na poschodie 1 a vysadí oboch pasažierov v čase t = 35.



Obrázok 3.7: Rozdiel medzi fungovaním klasického a predvídavého algoritmu

7. Backtrackingový algoritmus

Tento druh algoritmu predpokladá, že výťah má namiesto klasických privolávacích tlačidiel nahor a nadol možnosť zadať priamo poschodie, kam ide. Program predpokladá, že tak spraví každá osoba. Informáciu potom môže algoritmus rôzne spracovať. V tomto prípade si algoritmus vypočíta najrýchlejšiu cestu na obslúženie danej situácie. Robí to pomocou orezaného prehľadávania do hĺbky. Viac ako nasledujúcich osem poschodí dopredu nepočíta, pretože už to predstavuje $2^8 = 256$

rôznych možností ciest výťahu. Keď výťah dôjde na koniec vopred vypočítanej cesty a neobslúžil ešte všetkých ľudí, alebo sa do rady na výťah postaví nový človek, výťah si prepočíta novú cestu. Viac danú myšlienku rozoberá Benjamin Hiller[2].

3.5 Ostatné zaujímavosti programu

Aby program nemusel zakaždým prepočítavať najkratšiu cestu v budove, pretože nie vždy existuje priama cesta, bolo potrebné vytvoriť triedu cesty. V nej sa jedenkrát prepočítajú najkratšie cesty z každého poschodia na ostatné pomocou Floydovho-Warshallovho algoritmu na hľadanie najkratších ciest v grafoch. Budova predstavuje totiž graf, pričom hrany grafu sú cesty výťahu z jedného na iné poschodie. Odtiaľ si program vytiahne vždy konkrétny údaj, ako sa dostane z poschodia A na poschodie B najkratšou cestou.

Template trieda kalendár očakáva triedu, ktorá má preťažený operátor < a má metódu get_cas, ktorá vráti premennú triedy cas. Tá druhá podmienka je tam nutná, kvôli verejnej metóde testovaci_vypis_casov. V programe týmto podmienkam vyhovujú len dve triedy. Trieda udalost a grafika_udalost.

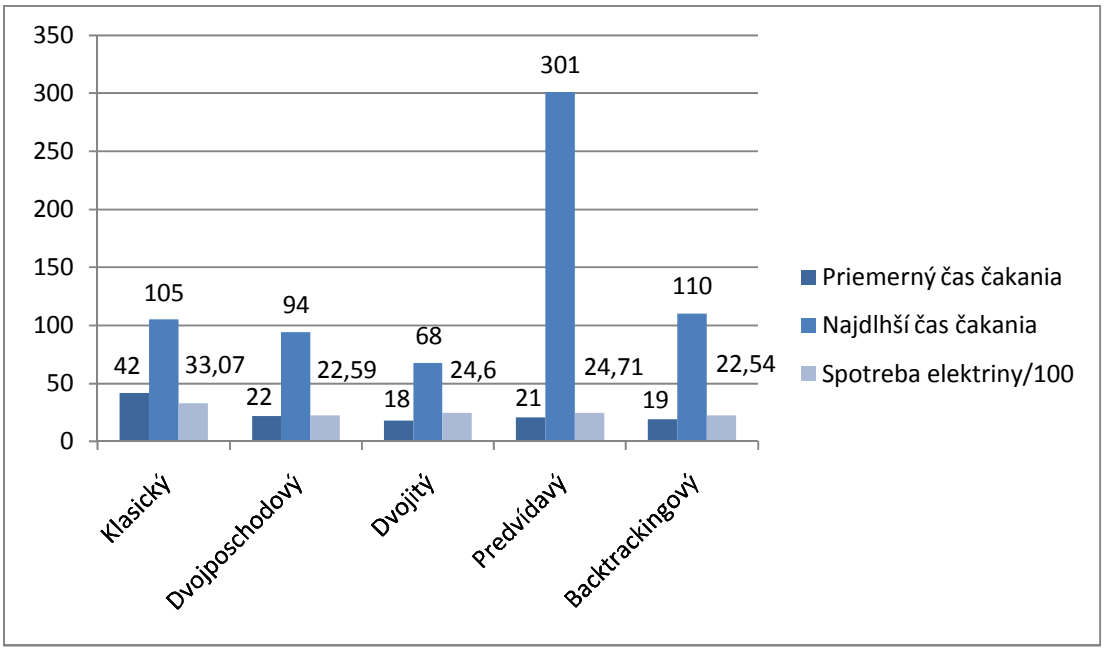
4 Praktické testy

4.1 Malé testy

Pomocou generátoru bola vytvorená sada vstupných súborov. Na nich sa testovali všetky algoritmy okrem dvoch, a to rýchleho – daný algoritmus je doplnkový a slúži pri vysokých budovách na rozdelenie budovy na viaceré časti, a viacposchodového – daný algoritmus je pre menej ako 17 poschodí zhodný s klasickým algoritmom.

Prvé testovanie

V prvom prípade sa najprv porovnávali jednotlivé algoritmy pre deväťposchodovú budovu, v ktorej sa nachádza iba jeden výťah. V prípade dvojitého algoritmu sa tam nachádzajú dva výťahy, vyplýva to z jeho definície. Bolo vygenerovaných 80 osôb, čo je približne po deväť osôb na poschodie.



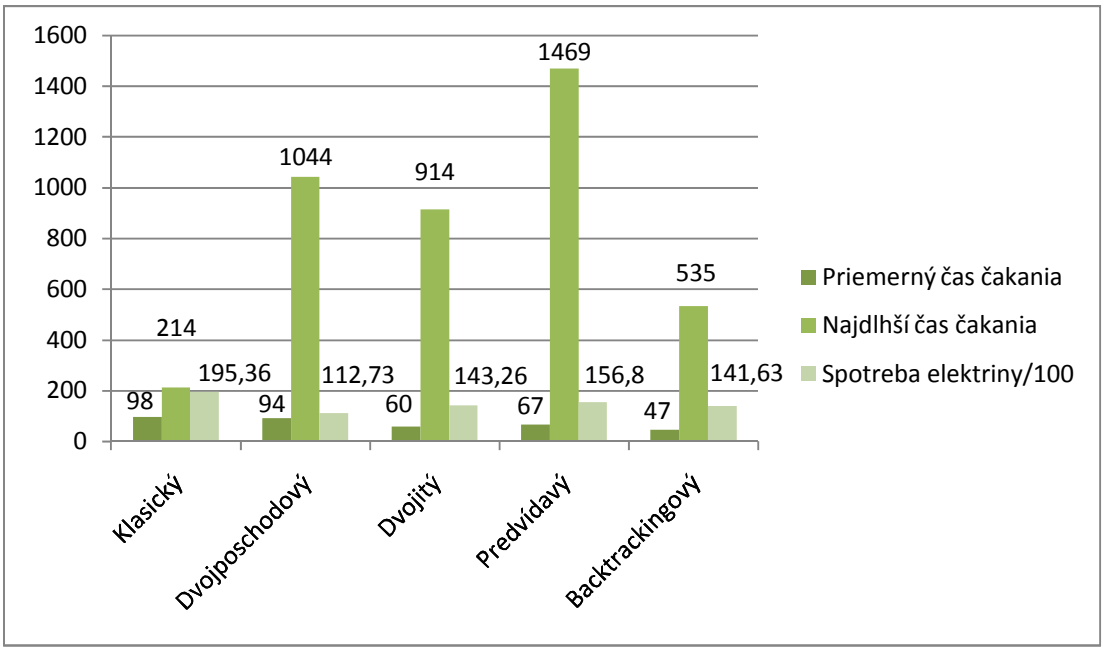
Obrázok 4.1: Štatistické údaje z prvého testovania

Na obrázku 4.1 vidíme štatistické údaje z prvej série testov. Ukazujú, že najkratšie sa čakalo na dvojitý výťah. Je to pochopiteľné, pretože daný výťah sú vlastne dva výťahy. Okrem tohto výťahu bol najkratší čas čakania na backtrackingový výťah. Vďaka malému množstvu prepravovaných osôb si výťah

prerátal najrýchlejšie obslúženie požiadaviek, a preto dosiahol najmenší priemerný čas čakania. Najdlhší čas čakania bol pri predvídavom výťahu, čo bol prekvapujúci výsledok testu. Pravdepodobne došlo k situácii, keď na viacerých poschodiach za sebou predvídavý výťah čakal dlhšiu dobu na prichádzajúcu osobu, čo spôsobilo dlhší čas čakania.

Druhé testovanie

Pomocou generátoru bola vytvorená ďalšia sada vstupných súborov. Tentoraz sa jednalo o dvadsaťjedenposchodovú budovu, v ktorej boli štyri výťahy. V prípade dvojposchodového a dvojitého výťahu sa použili len dva, pretože každý z nich má namiesto jednej kabínky dve. Bolo vygenerovaných o 120 ľudí viac, čiže 200, čo predstavuje priemerne po desať osôb na poschodie.

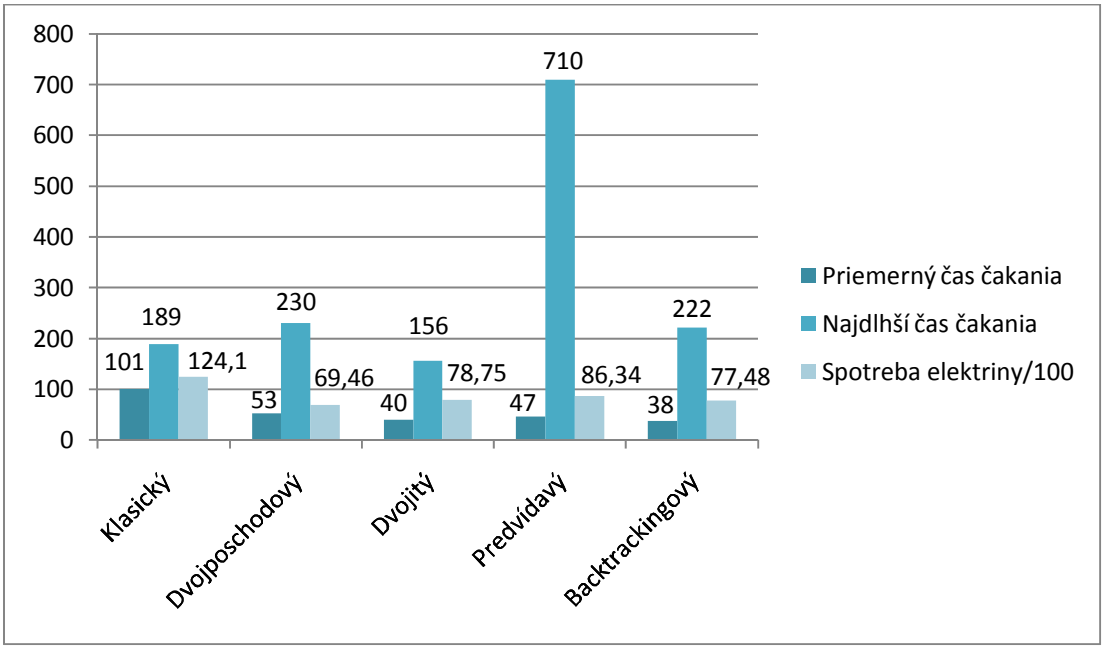


Obrázok 4.2: Štatistické údaje z druhého testovania

Obrázok 4.2 bol mierne prekvapujúci. Priemerné časy čakania boli vysoké. Asi to bolo spôsobené veľkým počtom osôb, ktorí žiadali o prepravu vo veľmi krátkych časoch. Z testu najlepšie vychádza backtrackingový výťah, ktorý má najkratší priemerný čas čakania. Taktiež má druhý najlepší výsledok najdlhšieho času čakania na výťah a druhú najmenšiu spotrebu výťahu. Predvídavý výťah má opäť prekvapivo veľmi zlý výsledok najdlhšieho času čakania.

Tretie testovanie

Na základe výsledkov z druhej série testov sa pre ďalšie testovanie vstupné dáta z druhého testovania len upravili. Konkrétne sa počet ľudí znížil pre danú budovu z 200 na 100, teda približne po päť osôb na poschodie. Ostatné dáta ostali nezmenené.

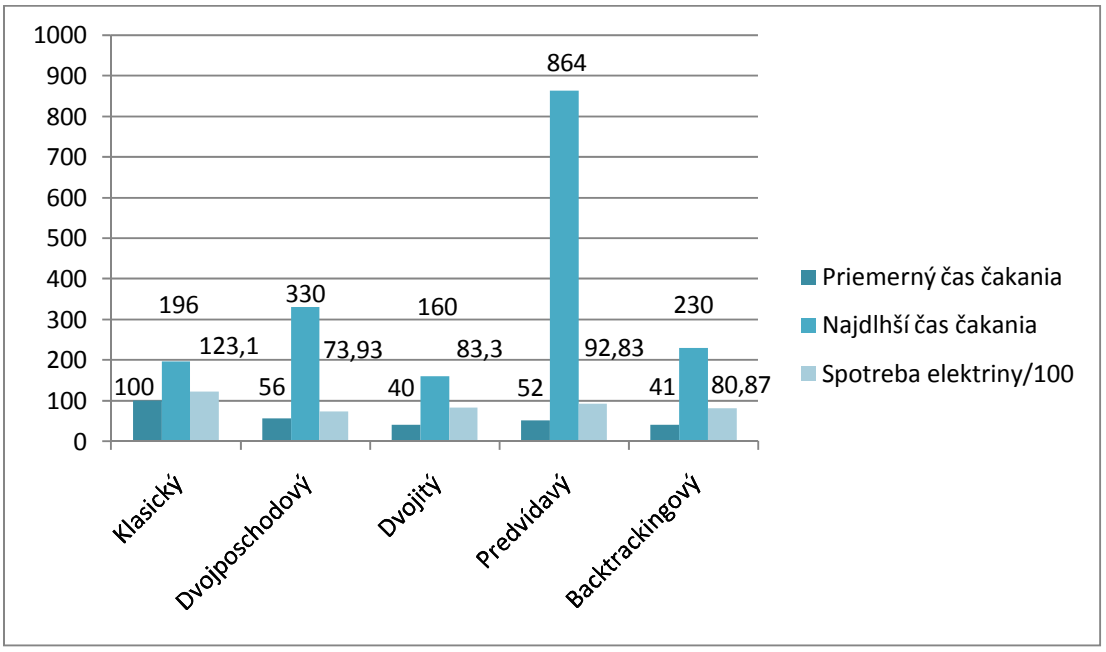


Obrázok 4.3: Štatistické údaje z tretieho testovania

Výsledok bol veľmi zaujímavý. Ukázalo sa, že iba jediný ukazovateľ sa zvýšil a ostatné sa znížili. Pri klasickom výťahu narástol priemerný čas čakania. Naopak znížili sa priemerné časy čakania pri ostatných výťahoch. Bol to očakávaný výsledok. Skrátil sa tiež čas najdlhšieho čakania pri jednotlivých výťahoch. Opäť bolo prekvapujúce, že najdlhší čas čakania nastal u predvídavého výťahu.

Štvrté testovanie

Pre štvrtý prípad bolo oproti predchádzajúcemu prípadu vygenerovaných 100 nových ľudí, aby sa porovnalo, že rôzne dáta nemajú veľký vplyv na výsledky testov. Ostatné dáta neboli oproti predchádzajúcemu testovaniu zmenené.



Obrázok 4.4: Štatistické údaje zo štvrtého testovania

Štvrté testovanie prinieslo očakávané výsledky. Až na jediný údaj sa poradie nezmenilo. Backtrackingový výťah sa dostal na druhé miesto v priemernom čase čakania. Predbehol ho dvojitý výťah. Ukazuje sa, že pri výsledkoch, ktoré sa líšia len minimálne, sa môže poradie s rôznymi vstupnými dátami zmeniť.

Zhrnutie

Na týchto štyroch testoch sa tiež ukázalo, že najviac spotrebúva elektrinu klasický výťah. Energeticky najvýhodnejšie vychádzajú dva výťahy, a to dvojposchodový a backtrackingový, u ktorých záleží, či je prevádzka veľká alebo malá. Ak je veľká, najlepšie výsledky má dvojposchodový výťah, ak je veľmi malá, o niečo výhodnejší sa javí backtrackingový výťah.

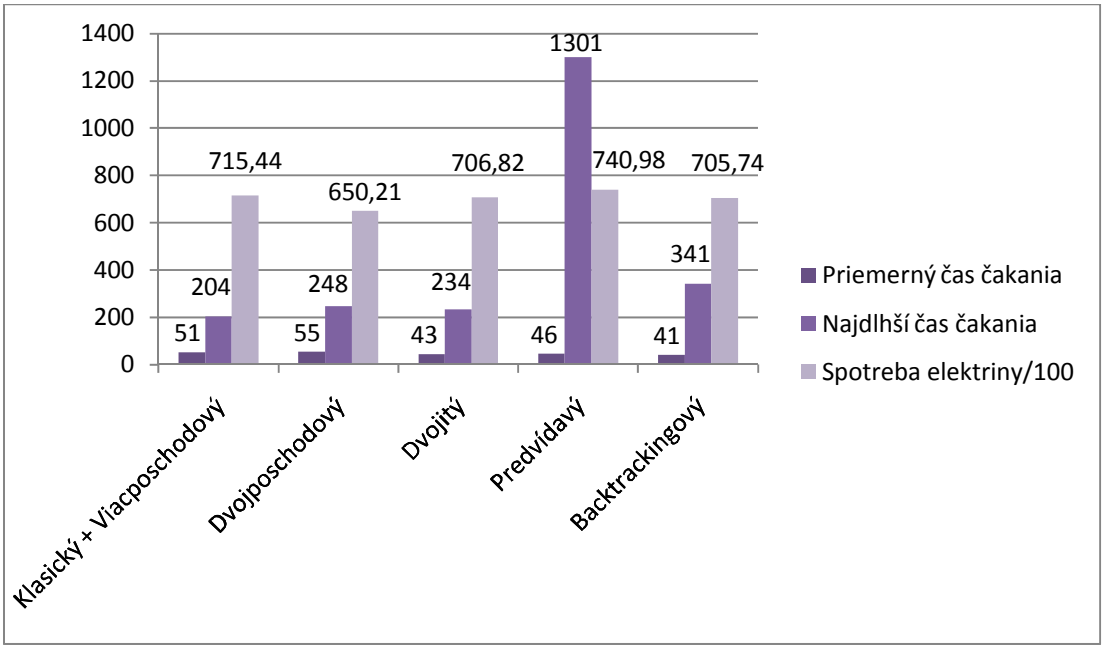
4.2 Veľké testy

Pre ďalšie testovanie boli použité rozsiahlejšie dáta. Konkrétne bola testovaná štyridsaťposchodová budova s 6 + 16 výťahmi. Prvých šesť výťahov bolo riadených rýchlym algoritmom a prepravovali osoby len na nulté a dvadsiate poschodia tzv. haly. Na ostatné poschodia sa osoby prepravovali ďalšími šestnástimi výťahmi po osem v každej polovici budovy. Tieto výťahy boli riadené ostatnými algoritmami.

V nasledujúcich grafoch sa časy čakania v prípade použitia viacerých výťahov jednou osobou počas jednej cesty nebudú sčítavať do kritéria priemerný čas čakania a najdlhší čas čakania.

Piate testovanie

Bolo vygenerovaných 400 osôb, čo je približne po desať osôb na poschodie. Prvýkrát bol použitý viacposchodový algoritmus, ktorým boli riadené v každej polovici budovy štyri výťahy. Ostatných osem výťahov bolo riadených klasickým algoritmom.

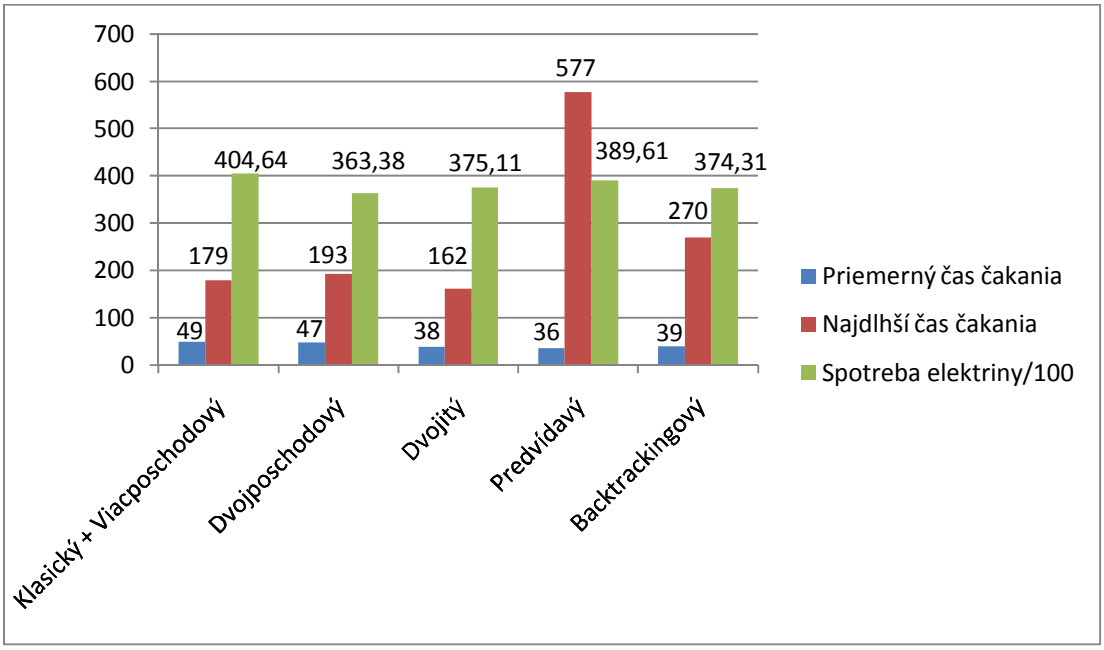


Obrázok 4.5: Štatistické údaje z piateho testovania

Na obrázku 4.5 vidíme výsledok. V daných výsledkov je zahrnutých aj šesť rýchlych výťahov, pretože sa podieľali na preprave osôb a vytvárajú celkový obraz návrhu výťahov v budove. Priemerné časy čakania boli u všetkých výťahov veľmi podobné. Najlepšie dopadol backtrackingový výťah. Najhoršie prekvapivo dvojposchodový výťah. Najhorší prípad čakania bol opäť u predvídavého výťahu, hoci priemerný čas čakania bol lepší ako u klasického. Najmenší rozdiel medzi priemerným čakaním a najdlhším čakaním dosiahol klasický výťah. Čo sa týka spotreby elektriny najlepšie dopadol dvojposchodový výťah a najhoršie predvídavý.

Šieste testovanie

Pre šiestu sadu testov bola upravená sada testov z piateho testovania. Konkrétne sa znížil počet ľudí na polovicu zo 400 na 200, čím sa znížilo zaťaženie na výťahy.



Obrázok 4.6: Štatistické údaje zo šiesteho testovania

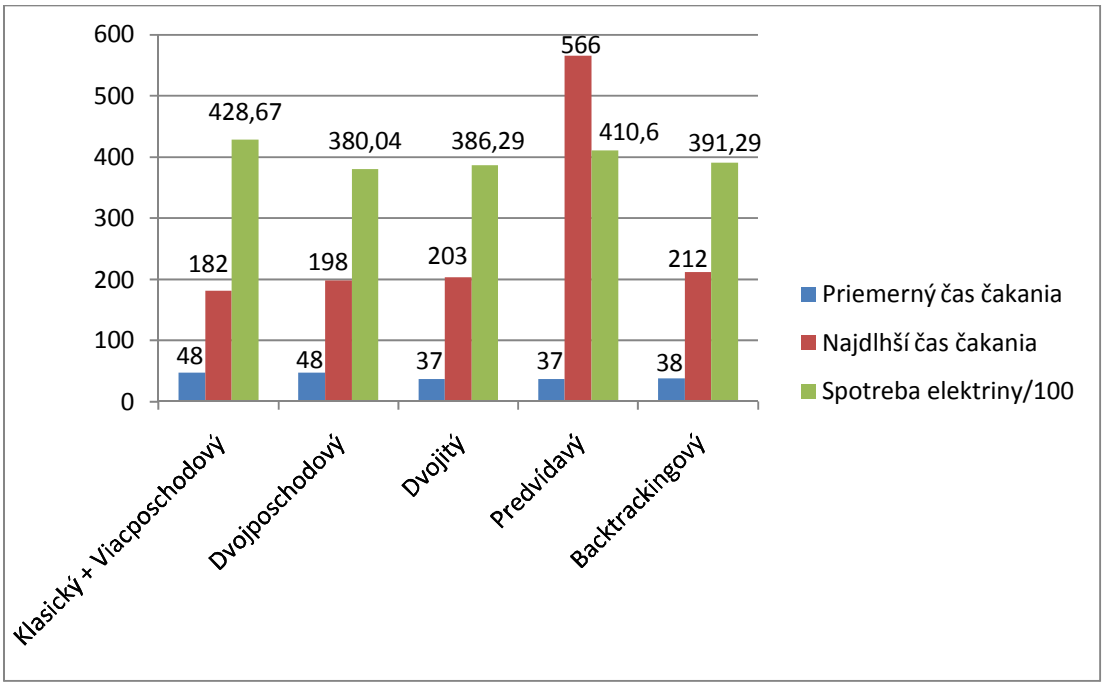
Obrázok 4.6 ukazuje zopár zmien oproti obrázku 4.5. Najlepší priemerný čas čakania dosiahol predvídavý výťah, hoci opäť dosiahol najhorší čas čakania. Backtrackingový výťah sa v tomto ohľade ocitol až na treťom mieste. Predbehol ho totiž dvojitý výťah. Pri ňom sa tiež dosiahol najlepší výsledok najdlhšieho času čakania. Čakalo sa naň najkratšie. Klasický výťah sa umiestnil na druhom mieste. V spotrebe elektriny opäť vyhral dvojposchodový výťah. Za ním sa dosť podobne umiestnili dvojitý a backtrackingový výťah.

Siedme testovanie

Pre siedmy prípad bolo oproti predchádzajúcemu prípadu vygenerovaných 200 nových ľudí. Ostatné dáta neboli zmenené. Cieľom bolo ukázať, že netreba robiť viacero testovaní pre rovnaký prípad.

Na obrázku 4.7 vidíme oproti predchádzajúcemu testovaniu jediné prekvapenie, a to dvojitý výťah. Dosiahol odlišné výsledky a umiestnenia vo všetkých kategóriách. V priemernom čase čakania sa zlepšil a vyrovnal sa

predvídavému výťahu, v najdlhšom čase čakania klesol z prvej na tretiu pozíciu. V spotrebe výťahu sa presunul z tretieho na druhé miesto a tým predbehol backtrakingový algoritmus. Inak sa ukazuje, že rôzne dáta nemajú veľký vplyv na konečné štatistické údaje. Z toho dôvodu stačí pre každé riešenie usporiadania výťahov vo výškovej budove jedna testovacia sada.

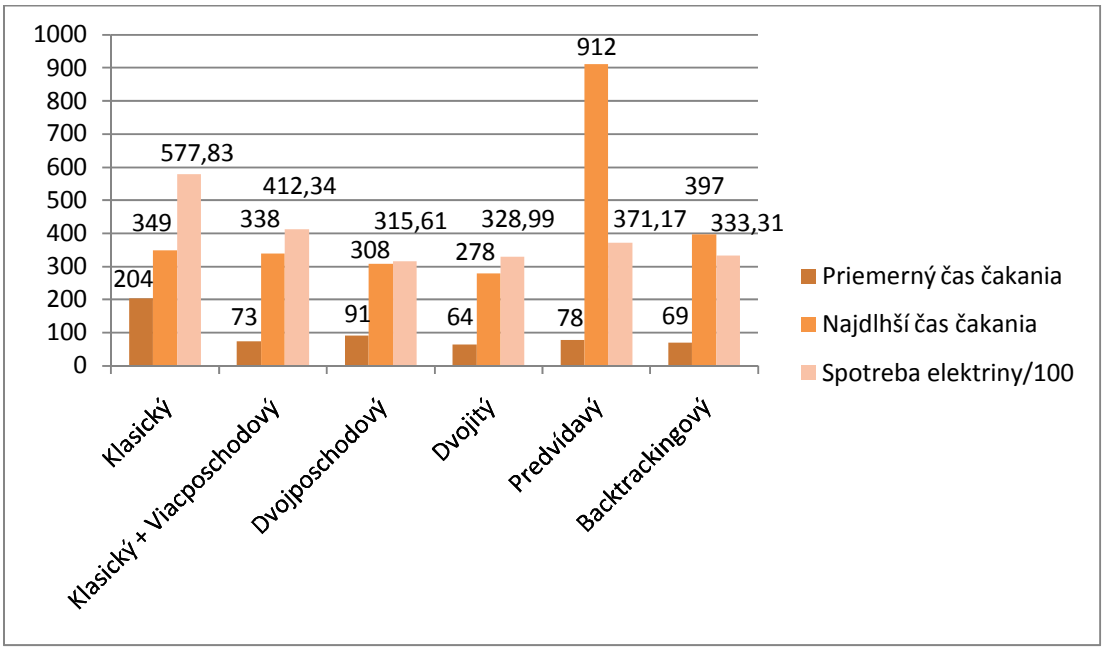


Obrázok 4.7: Štatistické údaje zo siedmeho testovania

Ôsme testovanie

Pre porovnanie bola vytvorená testovacia sada so štyridsaťposchodovou budovou, v ktorej sa zmenilo rozdelenie výťahov. Hala v strede budovy bola zrušená a všetky výťahy vychádzali z nultého poschodia. Počet výťahov a prepravovaných osôb sa nezmenil. Pre lepšie porovnanie výsledkov bola testovacia sada rozšírená o test s výťahmi riadenými klasickým algoritmom a o test, v ktorom sa kombinuje klasický a viacposchodový algoritmus.

Na obrázku 4.8 je vidieť, že je podstatný rozdiel, či sú v budove výťahy riadené čisto klasickým algoritmom, alebo či sú kombinované s viacposchodovým algoritmom. V prospech viacposchodového algoritmu hovorí zlepšenie o 131 sekúnd (vyše dve minúty) pri priemernom čase čakania, ako aj zníženie spotreby elektriny o pätinu.



Obrázok 4.8: Štatistické údaje z ôsmeho testovania

Zhrnutie

Pri porovnaní šiesteho a ôsmeho testovania je vidieť, že priemerné časy čakania a najdlhšie časy čakania sa v ôsmom testovaní oproti predchádzajúcemu približne zdvojnásobili. Z toho vyplýva, že ak veľa osôb v budove používa výťahy na prepravovanie v jednej časti budovy, oplatí sa v budove vytvoriť haly. V opačnom prípade vychádza lepšie nevytvoriť halové poschodia, pretože ak ide osoba z jednej časti budovy do inej, a nezačína ani nekončí v halových poschodiach, tak priemerný čas čakania je trikrát dlhší, než priemerný čas chodenia v rámci jednej časti budovy. To je viac ako v prípade riešenia bez halových poschodí. V prospech halových poschodí ešte hovorí šetrenie priestoru potrebného pre šachty výťahov. Šachty v jednotlivých častiach budovy môžu byť nad sebou, teda zaberú menej priestoru z plochy poschodí budovy.

Testy v budove, kde sa zmení algoritmus len pri jednom výťahu, by nemali význam, pretože by neukázali výrazný rozdiel medzi jednotlivými algoritmami. V niektorých prípadoch by kombinovanie algoritmov výťahov prinieslo lepšie výsledky.

Zaujímavá by bola tiež možnosť, čo by spôsobilo skombinovanie dvoch algoritmov do jedného. Napríklad predvídavého algoritmu s backtrackingovým. K tomu by však bolo potrebné program upraviť alebo vytvoriť nový.

5 Záver

Výsledky simulácie ukazujú, že každý zo skúmaných algoritmov výťahov má svoje opodstatnenie a nedá sa povedať, že by niektorý z nich bol univerzálne najlepší alebo najhorší. Niektoré algoritmy sú vhodnejšie pre špecifické situácie ako je dopravná špička, iné pre situácie, keď je málo poschodí, na ktoré výťah chodí, alebo kde je málo prepravovaných osôb. Výsledky tiež ukazujú, že existuje možnosť, v ktorej sa klasický algoritmus pre riadenie výťahu javí ako lepší než iné algoritmy, preto sa stále oplatí používať ho v praxi. Taktiež jeho cena je asi najnižšia, čo je asi tiež jeden z dôvodov, prečo má význam ho používať. Ukázalo sa, že dvojposchodový výťah nemusí byť lepší než klasický, ale jeho spotreba energie, a to že zaberá jednu šachtu namiesto dvoch, sú cenné vlastnosti, ktoré ho uprednostňujú do výškových budov. Ukázalo sa, že ak spojíme dva výťahy do jedného (dva klasické výťahy do dvojitého), dostaneme vo väčšine prípadov lepšie výsledky. Z toho vyplýva, že riadiť výťahy skupinovo obyčajne vedie k lepšiemu výsledku.

Prínosom práce je, že sa podarilo porovnať rôzne algoritmy na rôzne veľkých vstupoch. Riešenie problému bolo veľmi zaujímavé a myslím, že aj výsledky tejto práce sú prínosom.

V práci by sa dalo pokračovať vytvorením skupinových algoritmov výťahov alebo zmenením predpokladu, že ľudia sú slušní a použijú výťah, ktorý si privolali. Taktiež by sa práca mohla viac priblížiť realite počítaním zrýchlenia výťahu, maximálnej rýchlosti výťahu, a počítania času presnejšie než na sekundy. Čas výpočtu simulácie by však dramaticky stúpol.

Literatúra

[1] KWON, Ohhoon; BAHN, Hyokyung; KOH, Kern. A Context-Aware Elevator Scheduling System for Smart Apartment Buildings. *Lecture Notes in Computer Science*. 2007, Volume 4413, s. 362-372. Dostupný také z WWW: <http://www.springerlink.com/content/un2j005253856j77/fulltext.pdf>.

[2] HILLER, Benjamin. *Combinatorial Optimization at Work* [online]. 10/01/2009 [cit. 2011-06-29]. Online optimization: Elevator and vehicle scheduling. Dostupné z WWW: <http://co-at-work.zib.de/berlin2009/downloads/2009-10-01/2009-10-01-1100-BH-Online-Optimization.pdf>.

[3] Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-06-29]. Dostupné z WWW: <http://en.wikipedia.org/ >.

Obsah CD

Priložené CD k tejto práci obsahuje:

- Stiahnutá knižnica freeglut a zdrojové súbory sú v adresári **zdrojove subory**
- tabuľky ku grafom v práci sú v adresári **grafy**
- projekty vo Visual Studio 2008 a Code::Blocks v adresári **projekty**
- súbory vstupov a výstupov ku 4. kapitole v adresári **testy**
- spustiteľný program pre Windows v adresári **program**
- vytvorené alebo upravené obrázky použité v tejto práci v adresári **obrazky**
- túto prácu vo formáte pdf